Service Oriented Architecture (SOA) Implementation Framework for Satellite Mission Control System Software Design

GSAW2006

28th March 2006

Soon Hie Tan
K I Thimothy

Nanyang Technological University Singapore

Concept of Service Oriented Architecture (SOA)

- SOA is an architectural style which emphasizes well-defined, loosely coupled, coarse-grained, business centric, reusable shared services
- Services are well-defined encapsulations of business assets which are described using standard and network based interfaces
- SOA is a distributed computing environment in which services/components interact on a peer-peer basis using standardized interfaces
- SOA is an evolution of the component based architecture, interfaced based design (Object Oriented Design) and the Distributed Object System Design – the concept is therefore not new but it is rapidly emerging as the premier integration and architecture framework for today's complex, heterogeneous computing environment of the business enterprises

Service Oriented Architecture (SOA) Implementation

- No standard reference model for SOA yet
- SOA shares the main concept of
  - Services
  - Service Descriptions
  - Advertising and Discovery
  - Specification of an Associated Data Model
  - Use of Service Contract
- The objective of SOA design is flexible and effective interoperation of coarse grained services so as to facilitate the composition, orchestration, encapsulation and management of the resultant business applications
- The de-facto SOA implementation framework today is the document-centric, messaging, service based distributed computing system commonly known as XML Web Services

NANYANG TECHNOLOGICAL UNIVERSITY

Evolution of SOA Concept

- Monolithic Design
  - Relative unstructured procedural coding
- Structured & Object Oriented Design
  - Program units based on functionalities
- Client Server (Two-Tier) Design
  - Bundling of functionalities into two tiers
  - Concept of remoting and distributed design
- Distributed Object (Multi-Tier) Design
  - Distributed object design
  - Object interactions in heterogeneous environment
- Component Object Model Architecture
  - Aggregation of objects into logical components with well defined, strongly typed interface
- Service Oriented Architecture
  - Aggregations of components to provide reusable coarse grained business functionalities
  - Interactions of coarse grained services with less strongly typed but standard based interfaces for flexible and effective interoperations

NANYANG TECHNOLOGICAL UNIVERSITY

SOA Implementation Framework – The XML Web Services

- Generally SOA has a message oriented middleware system which supports a well organized work flow definition to facilitate service interoperation
- To interoperate between services from multiple enterprises, SOA needs to provide service level agreements and operational policies
- Typically the Enterprise Service Bus (ESB) of the SOA is a message broker with standard based messaging protocol to enable interoperability between the coarse grained services. The communication layer provides reliable synchronous/asynchronous secure messaging to allow the services to interoperate through a connection layer
- A specific SOA implementation can be found in the XML Web Services implementation framework
- XML Web Services use HTTP protocol and XML based SOAP messaging system to provide SOA using a standard service interface based on the Web Services Description Language (WSDL), Universal Description, Discovery and Integration (UDDI), electronic business XML (ebXML), Security Assertion Markup Language (SAML) and Web Service Security (WS-Security)

System Design Considerations in SOA Implementations

- Distributed object systems provide stateful/stateless, singleton/singlecall fine-grained components with strongly typed interfaces for each component
- Distributed objects are tightly coupled over synchronous connections (typically in intranet environment)
- Web Services provide stateless, singlecall coarse-grained applications with less strongly typed interfaces and late bindings for data
- Web Services operate over loosely coupled synchronous/asynchronous connections (typically over internet)
- Web Services are not suitable for real-time or near real-time applications due to performance and QoS considerations. Distributed object systems fit in more naturally in such applications
- Web Services are not necessary if there is no need to offer services to external parties or to compose and orchestrate business applications from services available from multiple enterprises over the network
- Interoperation (and hence system integration) is the prime design objective for Web Services but distributed object systems also strive to achieve interoperability between components
- Portability of both software systems assists in the solution of interoperation and integration problems with existing non-remotable and legacy systems

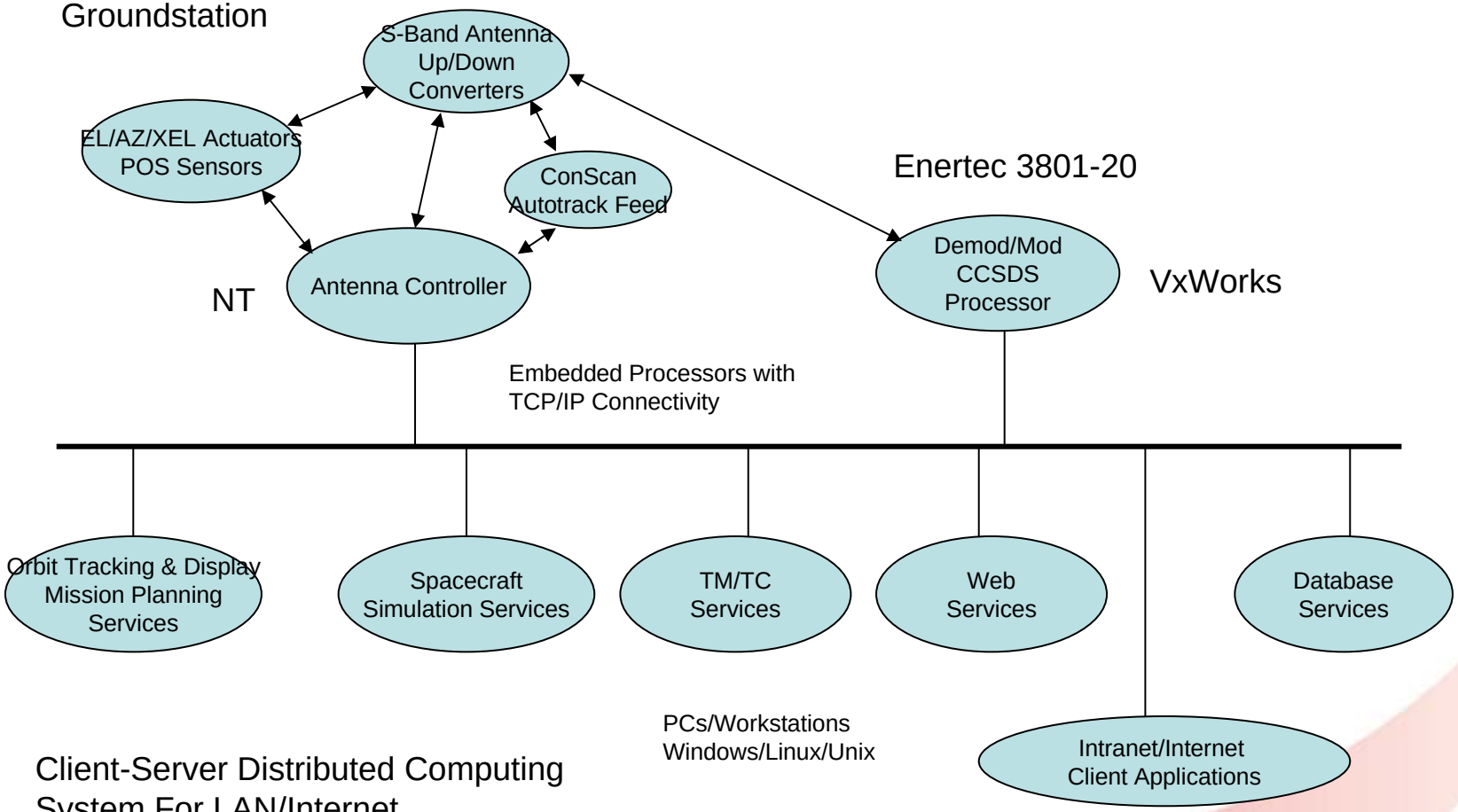
NANYANG TECHNOLOGICAL UNIVERSITY

X-Sat Mission Control System Design

- The satellite control software architecture incorporates the main concept of SOA to build an enterprise strength system for mission critical services
- Uses distributed computing architecture to build services using multiple middleware systems comprising
  - Tightly coupled distributed object systems for real-time or near real-time mission control services
  - Loosely coupled document-centric messaging systems (Web Services) for information technology (IT) services
- Assumes operation in a heterogeneous computing environment and additional interoperation requirements due to the use of COTS software packages, existing legacy information systems and embedded systems in ground system equipment

Ground System Architecture for TT&C

Integral Systems
Explorer 12000 6.1m
Groundstation



Client-Server Distributed Computing
System For LAN/Internet

NANYANG TECHNOLOGICAL UNIVERSITY

The Client Server model

- Business Logic Tier (Groundstation Services)
- Client Tier (GUI, Presentation)
- Back-end EIS Tier (COTS Packages, Database servers and Legacy Information Systems, Embedded processors)
- Platform used – Windows, Linux, Unix in PCs/Workstations

Middleware for Implementing Distributed Computing Systems

- Tightly coupled distributed object systems using following frameworks - .NET Remoting (C#), DCOM (C++, VB), J2SE RMI/IIOP (Java), J2SE IDL CORBA (Java), J2EE EJB (Java), ORBacus/OmniORB CORBA (C++, Java, Python)
- Loosely coupled document-centric messaging systems using .NET XML Web Services (C#) and J2EE based JAX-RPC XML Web Services (Java)
- The use of multiple middleware frameworks ensures vendor neutrality and long project life cycle by guarding against product obsolescence
- The software system design can capitalize on the strengths and features of the different distributed object frameworks so as to achieve ease of implementation and overall system performance.

NANYANG TECHNOLOGICAL UNIVERSITY

Types of Interoperations used in System Integration

- Out-of-Process Interoperation between different distributed computing systems across different platforms (interoperation across process and machine boundaries)
  - Distributed Object System interoperation (between .NET Remoting, J2SE RMI/IIOP, J2EE EJB, CORBA) is achieved by adding an IIOP channel to .NET Remoting
  - Web Services are able to interoperate due to their basic design for SOA
- In-Process Interoperation (interoperation within process boundary) between managed and unmanaged codes inside and outside of Virtual Machine Frameworks and In-Process Interoperation in other Native-OS Frameworks
- Interoperation with COTS packages (e.g. STK, Matlab Engine), Legacy systems, database servers & embedded processors in the EIS Tier based on out-of-process or in-process interoperation methods

Software System Portability

- In-Process Interoperation with non-remotable or legacy packages is greatly facilitated by the portability of the server tier software development framework
- “Write Once, Run Everywhere” – e.g. Java J2SE/J2EE, .NET (if the Mono Project for Linux/Unix is considered a usable framework compatible to .NET under Windows). Portability is enabled through the use of Virtual Machines (JVM, CLR) for the targeted platforms.
- “Write Once, Compile & Run Everywhere” – e.g. Qt & CORBA. Portability is enabled through source codes and compilers for use with the available framework for the targeted platforms.

Conclusions

- *Service/component interoperability* and *software system portability* are the key requirements for accomplishing the necessary system integration process used to compose and orchestrate business services
- Overall system design can achieve a service oriented architecture design concept at either the *service* or *component* levels of granularities, depending on the nature of the services
- This approach is more effective than the uniform use of the Web Services implementation framework for implementing all services and the use of adaptors for re-architecturing existing/legacy components to service level functionalities in order to conform with the generally accepted SOA design concepts.